TME 2 - Appariement

Exercice 1 – Exercice de compréhension : modèle de RI simple

On considère la collection de documents et la liste des stopwords du TME1. L'objectif dans cet exercice est d'estimer le score des documents pour la requête *"home sales top"*.

Q 1.1 On pensera à l'optimisation du calcul du score. Quels index faut-il interroger pour avoir un calcule du score pertinent ?

Q 1.2 Ecrire le code qui permet de calculer le score des documents à partir du modèle booléen.

Q 1.3 Ecrire le code qui permet de calculer le score des documents à partir du modèle vectoriel (produit scalaire) dans le cas d'une pondération tf.

Exercice 2 – Projet "Moteur de Recherche" : Etape Appariement

L'exercice précédent vous a permis de voir comment optimiser le calcul du score. a chaque appariement, il est nécessaire de définir quel est le modèle de RI utilisé ainsi que la façon de pondérer/représenter les termes dans les index. Dans le projet, nous allons séparer les classes permettant la pondération et celles définissant les modèles.

1 Représentation pondérée des documents et de la requête

Ecrire une classe générique `Weighter` qui a accès à l'objet `Index` et contenant au moins les méthodes suivantes :

- La méthode abstraite `getWeightsForDoc(idDoc)` qui retourne les poids des termes pour un document dont l'identifiant est `idDoc`.
- La méthode abstraite `getWeightsForStem(stem)` qui retourne les poids du terme `stem` pour tous les documents qui le contiennent.
- La méthode abstraite `getWeightsForQuery(query)` qui retourne les poids des termes de la requête.

Définir ensuite diverses classes de pondération qui héritent de `Weighter` correspondant aux schémas de pondération suivants :

- $w_{t,d} = tf_{t,d}$ et $w_{t,q} = 1$ si $t \in q$, O sinon ;
- $w_{t,d} = tf_{t,d}$ et $w_{t,q} = tf_{t,q}$
- $w_{t,d} = tf_{t,d}$ et $w_{t,q} = idf_t$ si $t \in q$, 0 sinon ;
- $w_{t,d} = 1 + log(tf_{t,d})$ si $t \in d$, 0 sinon ; et $w_{t,q} = idf_t$ si $t \in q$, 0
- $w_{t,d} = (1 + log(tf_{t,d}))x\ idf_t$ si $t \in d$, 0 sinon ; et $w_{t,q} = (1 + log(tf_{t,q}))x\ idf_t$ si $t \in q$, 0.

2 Modèles de RI

Q 2.1 Définir une classe `IRModel` qui a un accès à l'index et contient à minima les méthodes suivantes :

- La méthode abstraite `getScores(query)` qui retourne les scores des documents pour une requête.
- La méthode `getRanking(query)` qui retourne une liste de coupes (document-score) ordonnée par score décroissante.

Q 2.2 Définir une classe `Vectoriel` qui hérite de `IRModel`. Elle comporte en paramètres un `Weighter` défini dans l'étape précédente ainsi qu'un booléen `normalized` permettant de définir la fonction de score (produit

scalaire si faux et score cosinus si vrai).
Remarque : Pensez à optimiser votre code (e.g., les normes des vecteurs des documents ne doivent pas être calculés à chaque nouvelle requête.

Q 2.3 Définir les classes `ModeleLangue` et `Okapi` permettant de calculer les scores pour respectivement le modèle de langue (lissage Jelinek-Mercer) et Okapi-BM25.

3 Bonus - Très fortement conseillés

Pour les modèles précédents, vous avez fixé les valeurs des paramètres (e.g., λ pour le modèle de langue, $k1$ et b pour BM25). On se propose ici d'optimiser ces valeurs. On considère deux façons d'apprendre les paramètres.

Q 2.4 On sépare l'ensemble des requêtes en deux ensembles (train/test) et on définit une recherche de valeur optimale par GridSearch. 1) définir une grille de valeurs à tester (e.g., de 0 à 1 par pas de 0.1). 2) expérimenter chaque combinaison possible pour chaque modèle sur le jeu de données train. 3) on pourra tester dans le TME 3 la combinaison qui obtient la meilleure valeur de métrique (MAP, Précision, ...), 4) on applique ces valeurs sur le jeu de test

Q 2.5 On sépare le jeu de données en k folds et on procède à une cross validation. Chaque fold est successivement utilisé pour l'étape de test alors que les autres sont utilisés pour le train et apprendre les paramètres. Il y a donc autant de valeurs de paramètres que de fold. A la fin, on fait la moyenne sur les modèles obtenus sur l'ensemble des folds utilisés en tant que test... On en parle en cours ;)